Theoretical Computer Science 410 (2009) 2537–2548

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

State complexity of basic operations on suffix-free regular languages☆

Yo-Sub Han [a,*], Kai Salomaa [b]

[a] Department of Computer Science, Yonsei University, Seoul 120-749, Republic of Korea

[b] School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada

ARTICLE INFO

Article history:
Received 19 November 2007
Received in revised form 24 September 2008
Accepted 8 December 2008
Communicated by M. Ito

Keywords:
State complexity
Suffix-free regular languages

ABSTRACT

We investigate the state complexity of basic operations for suffix-free regular languages. The state complexity of an operation for regular languages is the number of states that are necessary and sufficient in the worst-case for the minimal deterministic finite-state automaton that accepts the language obtained from the operation. We establish the precise state complexity of catenation, Kleene star, reversal and the Boolean operations for suffix-free regular languages.

1. Introduction

Codes are useful in many areas such as information processing, data compression, cryptography and information transmission [15]. Some of the well-known codes are prefix codes, suffix codes, bifix codes and infix codes. People use different codes for different applications based on the characteristic of each code [1,15]. Since codes are sets of strings over an alphabet, they are closely related to formal languages: a code is a language. Thus, the condition defining a class of codes defines a corresponding subfamily of each language family. For regular languages, for example, suffix-freeness defines suffix-free regular languages, which constitute a subfamily of regular languages.

There are different ways to define the complexity of a regular language L. One classical definition is the total number of states in the minimal deterministic finite-state automaton (DFA) for L since the minimal DFA for L is unique (up to isomorphism) [12,20]. Based on this definition, Yu and his co-authors [23] defined the state complexity of an operation for regular languages to be the number of states that are necessary and sufficient in the worst-case for the minimal DFA that accepts the language obtained from the operation. Yu [22] gave a comprehensive survey of the state complexity of regular languages. Salomaa et al. [19] studied classes of languages for which the reversal operation reaches the exponential upper bound. As special cases of the state complexity, researchers examined the state complexity of finite languages [3,7], the state complexity of unary language operations [18] and the nondeterministic descriptional complexity of regular languages [10, 11]. There are several other results with respect to the state complexity of different operations [4–6,13,14,17].

Recently, Han et al. [8] examined the state complexity of prefix-free regular languages. They tackled the problem based on the structural property of prefix-free DFAs: A prefix-free DFA must be non-exiting assuming all states are useful [9]. It turns out that the state complexity for the prefix-free case is strictly less than the corresponding state complexity for regular languages over some basic operations. We know that if a language L is prefix-free, then its reversal L^R is suffix-free by definition. Moreover, if L is regular and non-empty, then the start state of a DFA for L^R should not have any in-transitions.

☆ A preliminary version appeared in Proceedings of the 32nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2007, LNCS, vol. 4708, Springer-Verlag, 2007, pp. 501–512.

* Corresponding author. Tel.: +82 2 958 5608; fax: +82 2 958 5649.
E-mail addresses: emmous@cs.yonsei.ac.kr (Y.-S. Han), ksalomaa@cs.queensu.ca (K. Salomaa).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.12.054

However, this condition is necessary but not sufficient. Due to this fact, the state complexity of suffix-free regular languages is not symmetric to the prefix-free case. This leads us to investigate the state complexity of basic operations on suffix-free regular languages. Interestingly, the results for catenation and Kleene star turn out to be of a totally different order from the case of prefix-free regular languages.

In Section 2, we define some basic notions. In Section 3, we examine the state complexity of Kleene star and reversal of suffix-free regular languages. We then look at the catenation of two suffix-free minimal DFAs in Section 4. Next, we investigate the state complexity of intersection and union of suffix-free regular languages based on the Cartesian product of states in Section 5. We present the comparison table of the state complexity on different types of regular languages and conclude the paper in Section 6.

2. Preliminaries

Let Σ denote a finite alphabet of characters and Σ^* denote the set of all strings over Σ. The size $|\Sigma|$ of Σ is the number of characters in Σ. A language over Σ is any subset of Σ^*. Given a set X, 2^X denotes the power set of X. For a string $x \in \Sigma^*$ and a character a, $|x|_a$ denotes the number of symbol a occurrences in x. We say that a string x is a *suffix* of a string y if $y = ux$ for some string u. We define a set X of strings to be a *suffix-free set* if a string from X is not a suffix of *any other* string in X. Given a string x from a set X, let x^R be the reversal of x, in which case $X^R = \{x^R \mid x \in X\}$.

The symbol $\emptyset$ denotes the empty language and the character λ denotes the empty string. A finite-state automaton (FA) A is specified by a tuple $(Q, \Sigma, \delta, s, F)$, where Q is a finite set of states, Σ is an input alphabet, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, $s \in Q$ is the start state and $F \subseteq Q$ is a set of final states. If F consists of a single state f, we use f instead of $\{f\}$ for simplicity. $|Q|$ denotes the number of states in Q. For a transition $q \in \delta(p, a)$ in A, we say that p has an *out-transition* and q has an *in-transition*. Furthermore, p is a *source state* of q and q is a *target state* of p. The transition function δ can be extended to a function $Q \times \Sigma^* \to 2^Q$ that reflects sequences of inputs. If $\delta(q, a)$ has a single element q', then we denote $\delta(q, a) = q'$ instead of $\delta(q, a) = \{q'\}$ for simplicity. A string x over Σ is accepted by A if there is a labeled path from s to a state in F such that this path spells out the string x. Namely, $\delta(s, x) \cap F \neq \emptyset$. The language $L(A)$ of an FA A is the set of all strings that are spelled out by paths from s to a final state in F. We say that A is *non-returning* if the start state of A does not have any in-transitions and A is *non-exiting* if all out-transitions of every final state in A go to the sink state. We assume that A has only *useful* states; namely, all states of A are reachable from the start state.

Given an FA $A = (Q, \Sigma, \delta, s, F)$, we define the *right language* L_q of a state q to be the set of strings that are spelled out by some path from q to a final state in A; namely, L_q is the language accepted by the FA obtained from A by changing the start state to q. We say that two states p and q are *equivalent* if $L_p = L_q$.

We define an FA A to be a DFA if the number of target states for each pair of a state q and a character $a \in \Sigma$ is one: namely, $|\delta(q, a)| = 1$. If A has m states, then we say that A is an m-state DFA. Given a DFA A, we define a state d to be a *sink state* if d is reachable from s of A and, for each $a \in \Sigma$, $\delta(d, a) = d$ and $d \notin F$. Since all sink states are always equivalent, we can assume that A has at most one sink state.

We define a (regular) language L to be suffix-free if L is a suffix-free set. A regular expression E is suffix-free if $L(E)$ is suffix-free. Similarly, an FA A is suffix-free if $L(A)$ is suffix-free. Moreover, if $L(A)$ is suffix-free and non-empty, then A must be non-returning. Similarly, we can define prefix-free regular expressions and languages. Note that if a language L is suffix-free, then L^R is prefix-free.

For complete background knowledge in automata theory, the reader may refer to textbooks [12,20].

3. Kleene star and reversal

Before examining the state complexity of various operations, we establish that any suffix-free (complete) minimal DFA must always have a sink state. Recall that the state complexity of a regular language L is the number of states in its minimal DFA. If L is a regular language, its minimal DFA does not necessarily have a sink state. However, if L is prefix-free, then its minimal DFA A must have a sink state since A is non-exiting. Therefore, we have to verify the existence of the sink state in a suffix-free minimal DFA before investigating the state complexity for each operation. This is crucial for computing the correct state complexity.

Lemma 1. *Let $A = (Q, \Sigma, \delta, s, F)$ be a minimal DFA for a suffix-free language and $k = |Q|$. Then, A has a sink state $d \in Q$ and for every string $w \in \Sigma^+$, $\delta(s, w^k) = d$.*

Proof. Let $w \in \Sigma^+$ be arbitrary and for the sake of contradiction assume that

$$\delta(s, w^k) \text{ is not a sink state.} \tag{1}$$

By the pigeon-hole principle, there exist $0 \leq j < m \leq k$ such that $\delta(s, w^j) = \delta(s, w^m)$ and let us denote this state by q. Now (1) implies that q is not a sink state and, since in a minimal DFA a final state is reachable from each state except for the sink state, there exists $v \in \Sigma^*$ such that $\delta(q, v) \in F$. Thus, A accepts both strings $w^j v$ and $w^m v$ and since $w \neq \lambda$ and $j \neq m$, this means that $L(A)$ cannot be suffix-free. ■

Lemma 1 shows that we must always consider the sink state for computing the state complexity of suffix-free regular languages. From now on, we assume that a suffix-free minimal DFA has the unique sink state.

3.1. Kleene star of suffix-free regular languages

We first start with the Kleene star operation.

Lemma 2. *Given an m-state suffix-free minimal DFA $A = (Q, \Sigma, \delta, s, F)$, $2^{m-2} + 1$ states are sufficient for $L(A)^*$, where $m \geq 2$.*

Proof. We first construct a nondeterministic finite-state automaton (NFA) A' from A that accepts $L(A)^*$ and determinize and minimize A'. Let d be the sink state of A. We compute $A' = (Q', \Sigma, \delta', s', F')$ as follows:

$$Q' = Q,$$
$$\delta'(p, a) = \begin{cases} \delta(p, a) & \text{if } p \notin F, \\ \delta(p, a) \cup \delta(s, a) & \text{if } p \in F. \end{cases}$$
$$s' = s,$$
$$F' = F \cup \{s'\}.$$

Note that the construction works because A is non-returning. It is easy to verify that A' accepts $L(A)^*$ from the construction. Note that the two sink states of A and A' are the same and A' is also non-returning. Now we apply the subset construction to A'. Let A_D denote the resulting DFA from A'. The number of states in A_D is 2^m. Note that a state of A_D is a subset of states in Q'. We identify equivalent states and merge them. We also remove unreachable states from the start state in A_D.

Claim 1. *Two states q_1 and q_2 of A_D are equivalent if $q_1 \setminus q_2 = d$ or $q_2 \setminus q_1 = d$.*

Proof: Assume that $q_1 \setminus q_2 = d$. (The other case is symmetry.) Since d is a sink state, it cannot appear in an accepting path in A_D and, therefore, $L_{q_1} = L_{q_2}$.

Since there are 2^{m-1} states that contain d in A_D, we can reduce 2^{m-1} states. Thus, the resulting DFA has $2^m - 2^{m-1} = 2^{m-1}$ states. Notice that the start state of A_D is $\{s'\}$ and a state of A_D is a subset of states in Q' from A'.

Claim 2. *A state q in A_D such that $s' \in q$ and $\{s'\} \neq q$ is not reachable from $\{s'\}$ in A_D since A' is non-returning.*

Based on **Claim 2**, we remove all states that contain s' and do not contain d in A_D except $\{s'\}$ itself. Therefore, we reduce $2^{m-2} - 1$ states. It shows that $2^{m-1} - (2^{m-2} - 1) = 2^{m-2} + 1$ states are sufficient for the minimal DFA of $L(A)^*$. ■

We now define a DFA A such that $L(A)$ is suffix-free and the state complexity of $L(A)^*$ reaches the upper bound in Lemma 2. Let $A = (Q, \Sigma, \delta, s, F)$, where $Q = \{0, 1, \ldots, m-1\}$, for $m \geq 4$, $\Sigma = \{a, b, c, d\}$, $s = m-2$, $F = \{0\}$ and δ is defined as follows:

(i) $\delta(m-2, c) = 0$,
(ii) $\delta(i, a) = i+1$, for $0 \leq i \leq m-4$, and $\delta(m-3, a) = 0$,
(iii) $\delta(i, d) = i$, for $1 \leq i \leq m-3$,
(iv) $\delta(m-2, b) = 1$, $\delta(0, b) = 0$, $\delta(i, b) = i$ for $2 \leq i \leq m-3$,
(v) all transitions not defined above go to the sink state $m-1$.

Fig. 1 depicts the DFA A. The figure omits the sink state $m-1$.

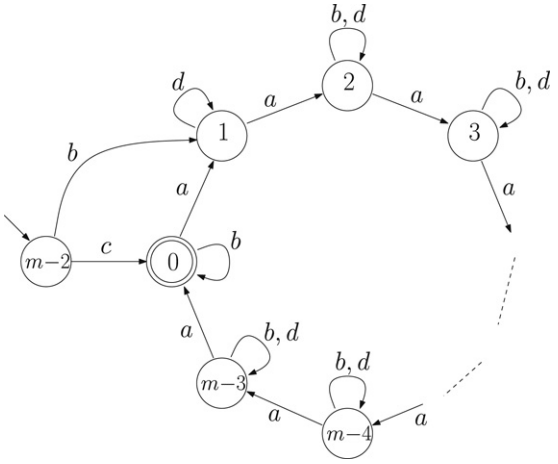

Fig. 1. The DFA A for the worst-case lower bound for the Kleene star of $L(A)$, for $m \geq 4$. Note that we omit the sink state $m-1$.

Lemma 3. *Let A be the DFA in Fig. 1 for $m \geq 4$.*

(1) *The language $L(A)$ is suffix-free.*

(2) *The state complexity of* $L(A)^*$ *is* $2^{m-2}+1$.

Proof. We prove two results separately.

(1) First, we prove that $L(A)$ is suffix-free. Assume that $L(A)$ is not suffix-free. This implies that there are two strings $w_1, w_2 \in L(A)$ such that

$$w_1 = uw_2, \quad u \neq \lambda.$$

Note that all strings of $L(A)$ begin with a symbol b or c, and c can occur only as the first symbol of any string. Since $u \neq \lambda$, we can write $w_2 = bw_2'$ for some $w_2' \in \Sigma^*$. Since we are at state 1 in A after reading the first b of bw_2', we know that

$$|w_2'|_a \equiv -1 \pmod{m-2}. \tag{2}$$

We consider the computation of A on $w_1 = ubw_2'$. Since A can reach the state 1 on input b only from the non-returning start state and $u \neq \lambda$, we know that A will not be in state 1 after reading the prefix ub. Now (2) implies that A cannot be in the final state 0 after reading w_1 – a contradiction. Therefore, $L(A)$ is suffix-free.

(2) Next, we prove that the state complexity of $L(A)^*$ is $2^{m-2}+1$.

Let $A' = (Q, \Sigma, \delta', s', F')$ be the NFA constructed for $L(A)^*$ as in the proof of Lemma 2 and let

$$A_D = (2^Q, \Sigma, \gamma, \{m-2\}, F_D)$$

be the equivalent DFA constructed from A' using the subset construction as in the proof of Lemma 2.

Consider the states in the minimal DFA for $L(A)^*$ from A_D. Let $\mathcal{C}$ be the collection of subsets of Q that consists of $\{m-2\}$ and all subsets of $\{0, 1, \ldots, m-3\}$. Namely,

$$\mathcal{C} = \{\{m-2\}\} \cup 2^{\{0,1,\ldots,m-3\}}.$$

Since $|\mathcal{C}| = 2^{m-2}+1$, it is sufficient to show that the two following claims are valid when regarded as states of A_D:

Claim 1. *All sets in* $\mathcal{C}$ *are pairwise inequivalent with respect to the right-invariant congruence of* $L(A)^*$.

Claim 2. *All sets in* $\mathcal{C}$ *are reachable in* A_D.

Proof of **Claim 1**: We first show that the singleton set $\{m-2\}$, which is the start state, is not equivalent with any subset of $\{0, 1, \ldots, m-3\}$. Let S be a subset of $\{0, 1, \ldots, m-3\}$.

- If S is non-empty, then there must be a state $i \in \{0, 1, \ldots, m-3\}$ such that $i \in S$. This implies that the string a^{m-2-i} is accepted from S but not from $\{m-2\}$.
- If S is empty, then c is accepted from $\{m-2\}$ but not from S.

Therefore, $\{m-2\}$ is not equivalent with any subset of $\{0, 1, \ldots, m-3\}$.

Next consider two subsets $S_1, S_2 \subseteq \{0, 1, \ldots, m-3\}$, $S_1 \neq S_2$. Without loss of generality, there exists $i \in S_1 \setminus S_2$ and $0 \leq i \leq m-3$, where the other possibility is completely symmetric. It follows that

$$0 \in \gamma(S_1, a^{m-2-i}) \quad \text{and} \quad 0 \notin \gamma(S_2, a^{m-2-i}).$$

Hence S_1 and S_2 are inequivalent.

Proof of **Claim 2**: We show that all sets in $\mathcal{C}$ are reachable. Since $\{m-2\}$ is the start state of A_D, it is reachable.

Using induction on k, for $0 \leq k \leq m-3$, we show that each of the sets

$$T_0 = \{0\}, \qquad T_k = \{0, m-3, m-4, \ldots, m-3-k+1\}, \quad 1 \leq k \leq m-3, \tag{3}$$

is reachable. Note that $\gamma(\{m-2\}, c) = \{0\} = T_0$. Due to A having a transition on b from the start state to 1, for $0 \leq k \leq m-4$,

$$\gamma(T_k, b) = T_k \cup \{1\}.$$

Note that $\delta(\{0\}, b) = \{1, m-2\}$. From this set, we get T_{k+1} by cycling with a-transitions, that is, $\gamma(T_k \cup \{1\}, a^{m-3}) = T_{k+1}$. We have shown that the set

$$T_{m-3} = \{0, 1, \ldots, m-3\}$$

is reachable. Since the d-transition is undefined in the state 0 (when we interpret transitions to the sink state to be undefined), and the d-transition is a self-loop for states $1, \ldots, m-3$, we observe that for any set $S \subseteq \{0, 1, \ldots, m-3\}$,

$$\gamma(S, d) = S \setminus \{0\}. \tag{4}$$

Now from the set T_{m-3}, we can reach any subset of $\{0, 1, \ldots, m-3\}$ by cycling each element, which we want to eliminate, to state 0 using the a-transitions and applying a d-transition as in (4). When we have the correct number of elements with correct intervals between consecutive elements, the set can be shifted cyclically to the correct position using only a-transitions. This concludes the proof of **Claim 2**.

Therefore, two results of Lemma 3 are true. ■

Combining Lemmas 2 and 3, we have the following result.

Theorem 4. *Given an* m*-state suffix-free minimal DFA* A, $2^{m-2}+1$ *states are necessary and sufficient in the worst-case for the minimal DFA of* $L(A)^*$, *where* $m \geq 2$.

Proof. We assume that $m \geq 4$ in Lemma 3. For $m = 3$, one can use $L(ba^*)$. Remark that any non-trivial suffix-free language needs at least 3 states (only $\{\lambda\}$ has two states). Therefore, the statement is true. ■

The proof of Lemma 3 uses a four character alphabet. It remains an open question whether the bound of Theorem 4 can be reached using an alphabet of size 2 or 3.

3.2. Reversal of suffix-free regular languages

We examine the reversal operation of suffix-free regular languages. First, we recall the state complexity of reversal on regular languages. If a regular language L is accepted by an m-state minimal DFA, then its reversal L^R is accepted by an m-state NFA. By the well-known subset argument, we can conclude that the state complexity of L^R is at most 2^m.

Proposition 5 (*Leiss [16] and Salomaa et al. [19]*). *There are classes of regular languages for which 2^m states are necessary and sufficient for the reversal of an m-state minimal DFA, where $m \geq 2$. Note that such an m-state minimal DFA does not have the sink state.*

Given a suffix-free minimal DFA $A = (Q, \Sigma, \delta, s, F)$, we flip all transition directions in A and obtain a new FA A^R for $L(A)^R$. If we apply the subset construction on A^R, then the resulting DFA is the minimal DFA for $L(A^R)$ [2,21].

Lemma 6. *Given an m-state suffix-free minimal DFA A, $2^{m-2}+1$ states are sufficient in the worst-case for the minimal DFA of $L(A)^R$.*

Proof. Let $A^R = (Q, \Sigma, \delta^R, F, s)$ be the FA obtained by flipping all transition directions in A and $m = |Q|$. Since A is non-returning, A^R is non-exiting. Before we carry out the subset construction, we remove useless states (a state is *useless* if it is not reachable from the start state in a given FA) from A^R. Let d be the sink state in A. Then, all states of $Q \setminus \{d\}$ in A^R do not have any out-transitions to d in δ^R. Namely, the sink state d is useless in A^R and, thus, we remove d and have $m-1$ states in A^R. Now we are ready for the subset construction. A crucial point of the subset construction is that the construction is based on all subsets of states of a given FA. There are 2^{m-1} subsets of states from A^R. Let $M' = (Q', \Sigma, \delta', s', F')$ be the resulting DFA by the subset construction from A^R. We examine a state of M', which is a subset of states from A^R, such that $s \in q$ and $\{s\} \neq q$.

Claim. *A state q in M' such that $s \in q$ and $\{s\} \neq q$ is unreachable from s' in M'.*

Proof: Assume that q is reachable for the sake of contradiction. Then, q is a final state of M' since $s \in q$. Since q is reachable, there is a path from s' to q in M' that reads a string x; namely, $x \in L(M')$. Let $p \neq s$ be another state in q. Since p is a useful state in A, there is a path from s to p in A and the path spells out a string $y \neq \lambda$. Then, in A^R, there is a path from p to s that spells out y^R. Therefore, there must be a path from q to a final state that spells out y^R in M'. This implies that M' accepts both x and xy^R. This contradicts the fact that $L(M')$ is prefix-free. Therefore, all such q are unreachable.

Our claim shows that there is only one final state $\{s\}$ in M', which is not a surprising result since $L(M')$ is prefix-free and the minimal DFA for a prefix-free regular language always has a single final state [9]. Thus, we remove all $2^{m-2}-1$ unreachable states. In summary, M' requires at most $2^{m-2}+1$ states. ■

Next, we show that $2^{m-2}+1$ states are necessary for the reversal of a suffix-free minimal DFA. Given a (regular) language L over Σ, $\#L$ is suffix-free if the character # is not in Σ.

We construct a suffix-free minimal DFA that has m states as follows: Let $A = (Q, \Sigma, \delta, s, F)$ be a minimal DFA as in Proposition 5 over Σ, which is not suffix-free in general. We introduce a new start state s' and a new transition $\delta(s', \#) = s$. We also introduce a sink state d that is appropriately connected. Note that a minimal DFA for a regular language in Proposition 5 does not have a sink state. Consequently, d is not equivalent with any of the states of A. Then, the new FA $A_\#$ is deterministic and minimal by construction. Furthermore, $L(A_\#)$ is suffix-free. Thus, if A has $m-2$ states, then $A_\#$ has m states. See Fig. 2 for an example.

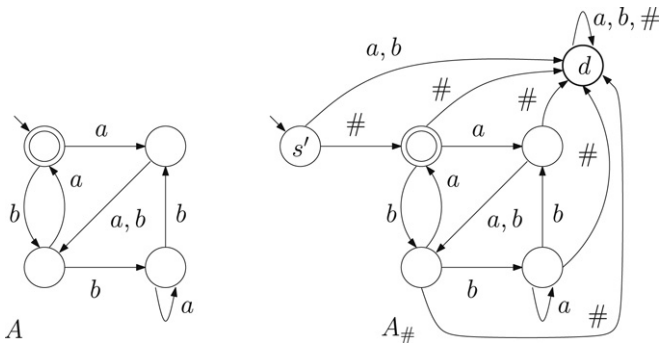

Fig. 2. An example of a minimal DFA A in Proposition 5. Note that $A_\#$ is also a minimal DFA and $L(A_\#)$ is suffix-free.

Lemma 7. *Given an m-state suffix-free minimal DFA $A_\#$ according to the described construction above, then $2^{m-2}+1$ states are necessary for the minimal DFA of $L(A_\#)^R$, where $m \geq 4$ and $\# \notin \Sigma$.*

Proof. Let A' be the minimal DFA for $L(A)^R$. Then, by Proposition 5, the state complexity of $L(A')$ is 2^{m-2} if A has $m-2$ states. Let F' be the set of final states of A'. We introduce a new state f'', which will be the only final state of the DFA A''. We connect all states in F' to f'' with label #. Let A'' denote the resulting DFA. Since A' has a sink state, we can make A'' complete using the sink state. Then, it is easy to verify that A'' is a minimal DFA and $L(A'') = L(A)^R \cdot \# = L(A_\#)^R$. Since A' has 2^{m-2} states, A'' has $2^{m-2}+1$ states. It follows that given an m-state suffix-free minimal DFA, $2^{m-2}+1$ states are necessary. ■

In the case $m = 3$ (respectively, $m = 2$), we can consider the suffix-free regular language $\{a\}$ (respectively, $\{\lambda\}$) whose reversal minimal DFA has three states (respectively, two states). All together with Lemmas 6 and 7, we establish the following theorem. Note that Salomaa et al. [19] established that the result of Proposition 5 holds also for binary alphabets.

Theorem 8. *Given an m-state suffix-free minimal DFA A over Σ, $2^{m-2}+1$ states are necessary and sufficient in the worst-case for the minimal DFA of $L(A)^R$, where $m \geq 2$ and $|\Sigma| \geq 3$.*

Note that if $m = 1$, then 1 state is sufficient and necessary since the reversal of the empty language is again the empty language.

4. Catenation

We investigate the state complexity of the catenation of two suffix-free regular languages. We first compute the upper bound and after that present a matching lower bound example.

Lemma 9. *Given two suffix-free minimal DFAs $A = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $B = (Q_2, \Sigma, \delta_2, s_2, F_2)$, then $(m-1)2^{n-2}+1$ states are sufficient for the minimal DFA of $L(A) \cdot L(B)$, where $m = |Q_1|$ and $n = |Q_2|$.*

Proof. Yu et al. [23] presented a DFA construction for the catenation of two DFAs. Based on their construction, we compute a DFA $C = (Q, \Sigma, \delta, s, F)$ for $L(A) \cdot L(B)$ as follows:

$Q = (Q_1 \times 2^{Q_2}) \setminus (F_1 \times 2^{Q_2 \setminus \{s_2\}})$, where 2^X denotes the power set of X,
$s = [s_1, \emptyset]$,
$F = \{[q, T] \in Q \mid T \cap F_2 \neq \emptyset\}$ and
$\delta([q, T], a) = [q', T']$ for $a \in \Sigma$, where

$$q' = \delta_1(q, a) \quad \text{and} \quad T' = \begin{cases} \delta_2(T, a) \cup \{s_2\} & \text{if } q' \in F_1. \\ \delta_2(T, a) & \text{otherwise.} \end{cases}$$

Note that $L(C) = L(A) \cdot L(B)$ and C is deterministic. Q is a set of pairs such that the first component of each pair is a state from Q_1 and the second component is a subset of Q_2. Q does not have pairs whose first component is a final state of A and whose second component does not contain s_2. Thus, the number of states in C is $m2^n - k2^{n-1}$ by construction, where $k = |F_1|$ is the number of final states in A. Now we minimize C and obtain the minimal DFA for $L(A) \cdot L(B)$.

Let d_2 denote the sink state of B. Due to d_2, two states $[q, P]$ and $[q, P \cup \{d_2\}]$ in C are equivalent, where $d_2 \notin P$. Thus, we merge all such states and reduce $(m2^{n-1} - k2^{n-2})$ states. Therefore, the number of remaining states is

$$m2^n - k2^{n-1} - (m2^{n-1} - k2^{n-2}) = m2^{n-1} - k2^{n-2}.$$

Next, we observe that $[s_1, T]$ is not reachable if $T \neq \emptyset$. This observation is valid since A is non-returning and, thus, s_1 has no in-transitions. It removes $(2^{n-1}-1)$ states and the current number of states is

$$m2^{n-1} - k2^{n-2} - (2^{n-1} - 1) = (m-1)2^{n-1} - k2^{n-2} + 1.$$

Lastly, we claim that $[q, T]$ is not reachable if $q \notin F_1$ and $s_2 \in T$. To prove this claim, we assume that $[q, T]$ is reachable for the sake of contradiction. It implies that there is a transition $\delta([q', T'], a) = [q, T]$ in C, where $[q', T']$ is also reachable and $a \in \Sigma$. It contradicts the fact that s_2 has no in-transitions in B. Thus, our claim is valid. The number of such states is $(m-1-k)2^{n-2}$. We remove these states and the total number of remaining states is

$$(m-1)2^{n-1} - k2^{n-2} + 1 - (m-1-k)2^{n-2} = (m-1)2^{n-2} + 1. \quad \blacksquare$$

We present two suffix-free minimal DFAs A and B such that the state complexity of $L(A)L(B)$ reaches the upper bound in Lemma 9. In the following, let $\Sigma = \{a, b, c, d\}$. We define

$$A = (Q_1, \Sigma, \delta_1, s_1, F_1), \tag{5}$$

where $Q_1 = \{0, 1, \ldots, m-1\}$, $m \geq 3$, $s_1 = 0$, $F_1 = \{1\}$ and δ_1 is defined as follows:

(i) $\delta_1(0, c) = 1$,
(ii) $\delta_1(i, a) = i + 1$, $1 \leq i \leq m-3$, $\delta_1(m-2, a) = 1$,
(iii) $\delta_1(i, b) = i$, $1 \leq i \leq m-2$,
(iv) $\delta_1(1, d) = 1$,
(v) all transitions not defined above go to the sink state $m-1$.

The DFA A is depicted in Fig. 3. The figure does not show the sink state $m-1$ or the transitions into the sink state.

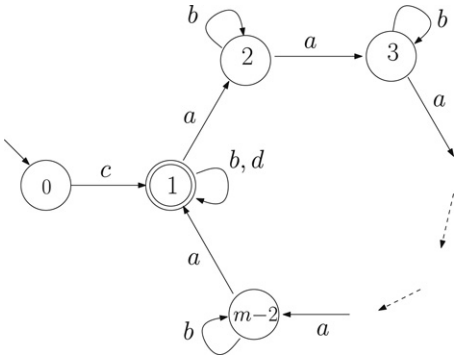

Fig. 3. The DFA A for the worst-case lower bound for catenation.

Next we define

$$B = (Q_2, \Sigma, \delta_2, s_2, F_2), \tag{6}$$

where $Q_2 = \{0, 1, \ldots, n-1\}$, $n \geq 3$, $s_2 = 0$, $F_2 = \{1\}$, and δ_2 is defined by the following:

(1) $\delta_2(0, d) = 1$,
(2) $\delta_2(i, b) = i + 1$, $1 \leq i \leq n-3$, $\delta_2(n-2, b) = 1$,
(3) $\delta_2(i, a) = \delta_2(i, c) = i$, $1 \leq i \leq n-2$,
(4) $\delta_2(i, d) = i$, $2 \leq i \leq n-2$,
(5) all transitions not defined above go to the sink state $n-1$.

The DFA B is depicted in Fig. 4. Again the figure does not show the sink state $n-1$.

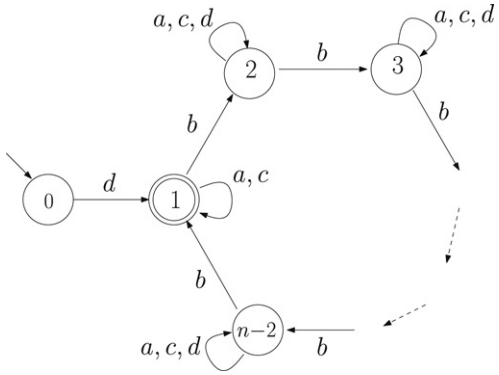

Fig. 4. The DFA B for the worst-case lower bound for catenation.

Lemma 10. *Let A be as in (5) and B as in (6), for $m, n \geq 3$.*

(1) *The languages $L(A)$ and $L(B)$ are suffix-free.*
(2) *The state complexity of $L(A) \cdot L(B)$ is $(m-1)2^{n-2} + 1$.*

Proof. (1) The language $L(A)$ is suffix-free because all strings accepted by A begin with a c and the symbol c may occur only as the first symbol in any strings of $L(A)$.

Now suppose that $w_1, w_2 \in L(B)$, where $w_1 = uw_2$, $u \neq \lambda$. All strings of $L(B)$ begin with a d and, hence, we can write $w_2 = dw_2'$. Since $w_2 \in L(B)$, we know that

$$|w_2'|_b \equiv 0 \pmod{n-2}. \tag{7}$$

This follows from the observation that in the cycle of B all transitions besides the b-transitions are self-loops except for the d-transition in state 1 that goes to the sink state. Consider now the computation of B on w_1. Since B is non-returning and the only way to reach state 1 with input symbol d is from state 0, it follows that after reading the prefix ud, the DFA B is in a state j, where $j \neq 0, 1$. Now (7) implies that the computation on w_2' starting from state j cannot end in the accepting state 1. This is a contradiction and we have shown that $L(B)$ is also suffix-free.

(2) By Lemma 9, it is sufficient to show that the state complexity of $L(A)L(B)$ is at least $(m-1)2^{n-2}+1$. Let $C = (Q, \Sigma, \delta, s, F)$ be the DFA constructed for $L(A)L(B)$ as in the proof of Lemma 9 (following the construction for catenation of arbitrary regular languages [23]).

We denote by $\mathcal{D}$ the subset of Q that consists of $[0, \emptyset]$, all elements $[i, T]$, for $i \in \{2, \ldots, m-1\}$ and $T \subseteq \{1, 2, \ldots, n-2\}$, and all elements $[1, T \cup \{0\}]$, for $T \subseteq \{1, 2, \ldots, n-2\}$. Since $\mathcal{D}$ has $(m-1)2^{n-2}+1$ elements, it is sufficient to establish the following two claims:

Claim 1. *All states of C belonging to $\mathcal{D}$ are pairwise inequivalent.*

Claim 2. *All elements of $\mathcal{D}$ are reachable as states of C.*

Proof of **Claim 1**: Consider an arbitrary state $[i, T] \in \mathcal{D}$, where $i \in \{1, 2, \ldots, m-1\}$ and $T \subseteq \{0, 1, 2, \ldots, n-2\}$. (Note that $0 \in T$ if and only if $i = 1$.) Now $\delta([i, T], c) = [m-1, T \setminus \{0\}]$ and, hence, $\delta([i, T], cd)$ cannot be an accepting state of C since the only d-transition that reaches the final state 1 is an out-transition of 0 in B. On the other hand, $\delta([0, \emptyset], cd) = [1, \{0, 1\}]$ and this shows that $[0, \emptyset]$ is not equivalent with any other state of $\mathcal{D}$.

Next, consider elements $[i, T], [j, T'] \in \mathcal{D}$, for $1 \le i < j \le m-1$ and $T, T' \subseteq \{0, 1, \ldots, n-2\}$. We note that in the computations of A, $\delta_1(i, a^{m-1-i}) = 1$ and $\delta_1(j, a^{m-1-i}) \ne 1$. This means that $\delta([i, T], a^{m-1-i}) = [1, R]$, $0 \in R$ and $\delta([j, T'], a^{m-1-i}) = [j', R']$, where $0 \notin R'$. Now because the only d-transition that reaches the final state 1 is an out-transition of 0 in B, we know that $\delta([i, T], a^{m-1-i}d) \in F$ and $\delta([j, T'], a^{m-1-i}d) \notin F$.

Finally, we consider elements $[i, T], [i, T'] \in \mathcal{D}$, for $1 \le i \le m-1$ and $T, T' \subseteq \{0, 1, \ldots, n-2\}$, $T \ne T'$. Without loss of generality, we can choose $k \in T \setminus T'$. We note that $k \ne 0$ since $0 \in T$ if and only if $i = 1$ (if $i = 1$, then $0 \in T'$). Since in B the b-transitions cycle the states $1, \ldots, n-2$ and take state 0 to the sink state, we observe that $\delta([i, T], b^{n-1-k}) \in F$ and $\delta([i, T'], b^{n-1-k}) \notin F$.

Proof of **Claim 2**: We show that all elements of $\mathcal{D}$ are reachable as states of C. First, $[0, \emptyset]$ is reachable as the start state of C.

Using induction on

$$k = |T \cap \{1, \ldots, n-2\}|$$

we show that any element $[i, T]$, $1 \le i \le m-1$, $T \subseteq \{0, 1, \ldots, n-2\}$ is reachable. Recall that for $[i, T] \in \mathcal{D}$, $0 \in T$ if and only if $i = 1$.

As the base case, consider $k = 0$. For $1 \le i \le m-2$, $\delta([0, \emptyset], ca^{i-1}) = [i, R]$ where $R = \emptyset$ if $2 \le i \le m-2$ and $R = \{0\}$ if $i = 1$. For $i = m-1$, we note that $\delta([0, \emptyset], cc) = [m-1, \emptyset]$.

Now consider an arbitrary element $[i, T]$, for $1 \le i \le m-1$, such that $T \cap \{1, \ldots, n-2\} = \{j_1, \ldots, j_{k+1}\}$ and $1 \le j_1 < \cdots < j_{k+1} \le n-2$ for $k \ge 0$. By the inductive assumption, the element

$$X = [m-2, \{j_2-(j_1-1), \ldots, j_{k+1}-(j_1-1)\}]$$

is reachable. Note that $2 \le j_r-(j_1-1) \le n-2$, for $r = 2, \ldots, k+1$, and hence the a- and d-transitions on these states are self-loops in B. Then,

$$\delta(X, ad) = [1, \{0, 1, j_2-(j_1-1), \ldots, j_{k+1}-(j_1-1)\}] = [1, Y],$$

and if $j_1 > 1$, then $\delta([1, Y], b^{j_1-1}) = [1, \{0, j_1, \ldots, j_{k+1}\}]$. From this state, we can reach $[i, T]$ with a^{i-1} if $1 \le i \le m-2$, and we can reach $[m-1, T]$ with symbol c. If $i > 1$, then $0 \notin T$.

Thus, we have shown that all elements of $\mathcal{D} \subseteq Q$ are reachable and, therefore, **Claim 2** is true.

This concludes the proof. ■

Lemma 10 shows that the upper bound in Lemma 9 is tight when $|\Sigma| \ge 4$.

Theorem 11. *For arbitrary $m, n \ge 3$, $(m-1)2^{n-2}+1$ states are necessary and sufficient in the worst-case for the catenation of, respectively, an m-state and an n-state suffix-free minimal DFAs.*

The worst-case example in Lemma 10 uses an alphabet with 4 characters. We do not know whether the upper bound can be reached using an alphabet of size 2 or 3.

5. Intersection and union

Note that for the complement operation of an m-state suffix-free DFA, it is easy to verify that m states are necessary and sufficient. In the following, we consider the operations of intersection and union.

5.1. Intersection of suffix-free regular languages

Given two DFAs A and B, we can construct a DFA for the intersection of $L(A)$ and $L(B)$ based on the Cartesian product of states. For details on the Cartesian product construction, refer to Hopcroft and Ullman [12].

Proposition 12. *Given two DFAs* $A = (Q_1, \Sigma, \delta_1, s_1, F_1)$ *and* $B = (Q_2, \Sigma, \delta_2, s_2, F_2)$, *let* $A \cap_c B = (Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F_1 \times F_2)$, *where for all* $p \in Q_1$ *and* $q \in Q_2$ *and* $a \in \Sigma$,

$$\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a)).$$

Then, $L(A \cap_c B) = L(A) \cap L(B)$.

Since the automaton $A \cap_c B$ constructed in Proposition 12 is deterministic, it follows that mn states are sufficient for the intersection of $L(A)$ and $L(B)$, where $|A| = m$ and $|B| = n$. Note that mn is a tight bound for the intersection of two regular languages [23].

We assign a unique number for each state from 1 to m in A and from 1 to n in B, where $|A| = m$ and $|B| = n$. Assume that the mth state and the nth state are the sink states in A and B, respectively. Let $A \cap_c B$ denote the resulting intersection automaton that we compute using the Cartesian product of states. By the construction, $A \cap_c B$ is deterministic since A and B are deterministic. Therefore, we obtain a DFA for $L(A) \cap L(B)$. Next, we minimize $A \cap_c B$ by merging all equivalent states and removing unreachable states from the start state.

Proposition 13 (*Han et al. [8]*). *For a state* (i, j) *in* $A \cap_c B$, *the right language* $L_{(i,j)}$ *of* (i, j) *is the intersection of* L_i *in* A *and* L_j *in* B.

Since a suffix-free DFA A has the sink state as proved in Lemma 1, $L_{(i,n)} = \emptyset$, for $1 \leq i \leq m$, by Proposition 13, where n is the sink state of B. Therefore, we can merge all these states. Similarly, all states (m, j), for $1 \leq j \leq n$, of $A \cap_c B$ are equivalent and, therefore, can be merged.

Observation 14. *Given suffix-free minimal DFAs A and B, all states* (i, n) *for* $1 \leq i \leq m$ *and all states* (m, j) *for* $1 \leq j \leq n$ *of* $A \cap_c B$ *are equivalent.*

Consider all states $(1, j)$, for $1 < j \leq n$, of $A \cap_c B$. Since $L(A)$ is suffix-free, the start state of A has no in-transitions. It implies that $(1, j)$ is not reachable from $(1, 1)$ in $A \cap_c B$ and, therefore, these states are useless as shown in Fig. 5. We can establish a similar result for the states $(i, 1)$, for $1 < i \leq m$.

Observation 15. *Given suffix-free minimal DFAs A and B, all states* $(i, 1)$, *for* $1 < i \leq m$, *and all states* $(1, j)$, *for* $1 < j \leq n$, *are useless in* $A \cap_c B$.

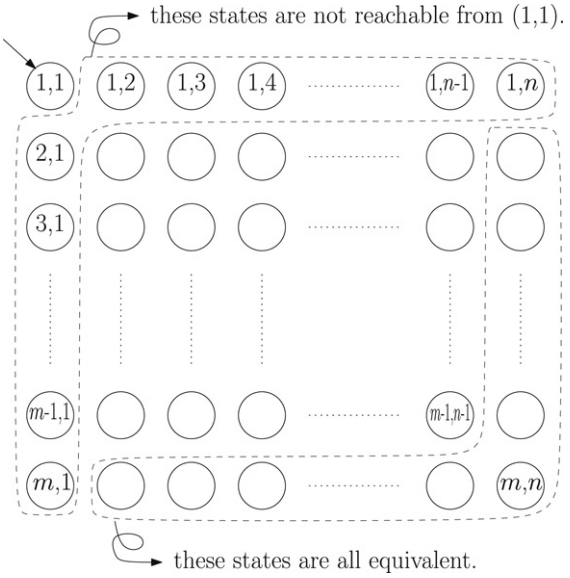

Fig. 5. The figure depicts the intersection automaton $A \cap_c B$ constructed for two suffix-free minimal DFAs A and B. Note that, by Observation 14, all states in the last row and in the last column are equivalent. Similarly, by Observation 15, all states, except for the start state $(1, 1)$, in the first row and in the first column are unreachable from $(1, 1)$.

Once we minimize $A \cap_c B$ based on Observations 14 and 15, the resulting minimal DFA has $mn - 2(m + n) + 6$ states.

Theorem 16. *Given two suffix-free minimal DFAs A and B,* $mn - 2(m+n) + 6$ *states are necessary and sufficient in the worst-case for the minimal DFA of* $L(A) \cap L(B)$, *where* $|\Sigma| \geq 3$ *and* $m, n \geq 3$.

Proof. The previous consideration together with Fig. 5 shows that $mn - 2(m+n) + 6$ states are sufficient.

We prove the necessary condition by giving two suffix-free minimal DFAs that reach the bound.

Assume that $\Sigma = \{a, b, \#\}$. Let A be the minimal DFA for

$$L = \{\#w \mid w \in \{a, b\}^*, |w|_a \equiv 0 \pmod{m-2}\}$$

and B be the minimal DFA for

$$L = \{\#w \mid w \in \{a, b\}^*, |w|_b \equiv 0 \pmod{n-2}\}$$

$L(A)$ and $L(B)$ are suffix-free since all strings have only one occurrence of # which may occur only as the first symbol in any string. It is easy to verify that $|A| = m$ and $|B| = n$. Let $L = L(A) \cap L(B)$. We claim that the minimal DFA for L needs $mn - 2(m+n) + 6$ states. To prove the claim, it is sufficient to show that there exist a set R of $mn - 2(m+n) + 6$ strings over Σ that are pairwise inequivalent modulo the right-invariant congruence of L.

Let $R = R_1 \cup R_2$, where

$R_1 = \{\lambda, \#\#\}$,
$R_2 = \{\#a^i b^j \mid 1 \le i \le m-2 \text{ and } 1 \le j \le n-2\}$.

Any string $\#a^i b^j$ from R_2 is inequivalent with λ since $\#a^i b^j \cdot \# \notin L$ but $\lambda \cdot \# \in L$. Similarly, $\#a^i b^j$ is inequivalent with $\#\#$ since $\#a^i b^j \cdot a^{m-2-i} b^{n-2-j} \in L$ but $\#\# \cdot a^{m-2-i} b^{n-2-j} \notin L$. The two strings λ and $\#\#$ of R_1 are inequivalent as well.

Next, consider two distinct strings $\#a^i b^j$ and $\#a^k b^l$ from R_2. Since $\#a^i b^j \neq \#a^k b^l$, $\#a^i b^j \cdot a^{m-2-i} b^{n-2-j} \in L$ but $\#a^k b^l \cdot a^{m-2-i} b^{n-2-j} \notin L$. Therefore, any two distinct strings from R_2 are inequivalent.

Thus, all $mn - 2(m+n) + 6$ strings in R are pairwise inequivalent. This concludes the proof. ■

5.2. Union of suffix-free regular languages

We now investigate the union of two suffix-free regular languages. We compute the union DFA for $L(A)$ and $L(B)$ using the Cartesian product of states. Given two suffix-free minimal DFAs $A = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $B = (Q_2, \Sigma, \delta_2, s_2, F_2)$, let $A \cup_c B = (Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F)$, where for all $p \in Q_1$ and $q \in Q_2$ and $a \in \Sigma$,

$$\delta((p, q), a) = (\delta(p, a), \delta(q, a))$$

and $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$. Then, $L(A \cup_c B) = L(A) \cup L(B)$ and $A \cup_c B$ is deterministic. Consider the right language of a state (i, j) in $A \cup_c B$.

Proposition 17 (*Han et al. [8]*). *For a state (i, j) in $A \cup_c B$, the right language $L_{(i,j)}$ of (i, j) is the union of L_i in A and L_j in B.*

Note that the two constructions for $A \cap_c B$ and $A \cup_c B$ are different. This implies that we may not be able to apply the same approach that we used for $A \cap_c B$ for computing the upper bound for $L(A) \cup L(B)$. For example, since $L_{(m,j)} = L_m \cup L_j \neq \emptyset$ by Proposition 17, all states (i, n) and (m, j) for $1 \le i \le m$ and $1 \le j \le n$, in $A \cup_c B$ are not necessarily equivalent. Thus, these states cannot be merged. On the other hand, we observe that all states $(i, 1)$ and $(1, j)$, for $1 < i \le m$ and $1 < j \le n$, are useless since $L(A)$ and $L(B)$ are suffix-free. Therefore, we minimize $A \cup_c B$ by removing these $m + n - 2$ states.

Theorem 18. *Given two suffix-free minimal DFAs A and B, $mn - (m+n) + 2$ states are necessary and sufficient in the worst-case for the minimal DFA of $L(A) \cup L(B)$, where $|\Sigma| \ge 5$ and $m, n \ge 3$.*

Proof. Let $A \cup_c B$ be the resulting DFA for $L(A) \cup L(B)$ by the Cartesian product of states. Since we can remove all states $(i, 1)$ and $(1, j)$, for $1 < i \le m$ and $1 < j \le n$, it is clear that $mn - (m+n) + 2$ states are sufficient.

We present two suffix-free minimal DFAs whose union reaches the bound. Let A be the minimal DFA for

$$\{\#w \mid w \in \{a, b, c\}^*, |w|_a \equiv 0 \pmod{m-2}\}$$

and B be the minimal DFA for

$$\{\#w \mid w \in \{a, c, d\}^*, |w|_c \equiv 0 \pmod{n-2}\}$$

over $\Sigma = \{\#, a, b, c, d\}$.

A has m states, which include one state to read the initial # and one sink state, and all strings of $L(A)$ do not have d's. Similarly, B has n states and all strings of $L(B)$ do not have b's. We argue that the minimal DFA for $L = L(A) \cup L(B)$ requires $mn - (m+n) + 2$ states. Let R be the set consisting of λ, $\#\#$ and the following strings:

$$\#a^i c^j, \quad \text{for } 0 \le i \le m-3 \text{ and } 0 \le j \le n-3, \tag{8}$$
$$\#ba^i, \quad \text{for } 0 \le i \le m-3, \tag{9}$$
$$\#dc^i, \quad \text{for } 0 \le i \le n-3. \tag{10}$$

We show that all strings of R are pairwise inequivalent in the right-invariant congruence defined by the language L.

(1) λ is not equivalent with any other string of R since λ followed by # can produce a string of L. However, this property does not hold for any other string of R.
(2) ## is not equivalent with any other string of R since no string can complete ## to be a string of L. However, this property does not hold for any other string of R.
(3) Consider two strings in (8): $\#a^xc^y$ and $\#a^sc^t$, where $(x, y) \neq (s, t)$. If $x \neq s$ ($y \neq t$ case is symmetric), then $\#a^xc^y \cdot ba^{m-2-s} \notin L$ but $\#a^sc^t \cdot ba^{m-2-s} \in L$. Therefore, no two strings of (8) are equivalent.
(4) Similarly to the third case, no two strings of (9) are equivalent and no two strings of (10) are equivalent.
(5) We show that any string of (8) cannot be equivalent with a string of (9): Consider $\#a^xc^y$ and $\#ba^z$ for $0 \leq x, z \leq m-3$ and $0 \leq y \leq n-3$. Now $\#a^xc^y(c^{n-2-y}d) \in L$ but $\#ba^z(c^{n-2-y}d) \notin L$ since no string in L can have both b and d.
(6) Any string of (9) cannot be equivalent with a string of (10): Consider $\#ba^i$ and $\#dc^j$. Now $\#ba^i(ba^{m-2-i}) \in L$ but $\#dc^j(ba^{m-2-i}) \notin L$.
(7) Comparing a string of (8) and a string of (10) is analogous to the sixth case.

Thus, the number of distinct equivalence classes of the minimal DFA for L is at least the cardinality of the set R. R (as defined above) has

$$2 + (m-2)(n-2) + (m-2) + (n-2) = mn - (m+n) + 2$$

elements. Therefore, $mn - (m+n) + 2$ states are necessary. ■

6. Conclusions

The state complexity of an operation for regular languages is the number of states that are necessary and sufficient for the minimal DFA that accepts the language obtained from the operation. Yu et al. [23] studied the operational state complexity of general regular languages and Han et al. [8] examined the state complexity of basic operations on prefix-free regular languages. Since suffix-freeness is the reversal of prefix-freeness, it was a natural problem to examine the state complexity of basic operations on suffix-free regular languages.

Based on the structural property that a suffix-free minimal DFA must be non-returning, we have tackled Kleene star, reversal, catenation, intersection and union cases and obtained the tight bound for each operation.

operation	regular languages	prefix-free case	suffix-free case
L_1^*	$2^{m-1} + 2^{m-2}$	m	$2^{m-2} + 1$
L_1^R	2^m	$2^{m-2} + 1$	$2^{m-2} + 1$
$L_1 \cdot L_2$	$(2m-1)2^{n-1}$	$m + n - 2$	$(m-1)2^{n-2} + 1$
$L_1 \cap L_2$	mn	$mn - 2(m+n) + 6$	$mn - 2(m+n) + 6$
$L_1 \cup L_2$	mn	$mn - 2$	$mn - (m+n) + 2$

Fig. 6. Operational state complexity of general, prefix-free and suffix-free regular languages.

Fig. 6 shows the comparison table of the state complexity on regular languages, prefix-free regular languages and suffix-free regular languages. We have established the tight state complexity bounds for each of the operations using languages over a fixed alphabet. However, the constructions usually require an alphabet of size 3 or 4 and, then, for most operations, it is open whether or not the upper bound for the state complexity of each operation can be reached using a small size alphabet such as $|\Sigma| = 2$ or 3.

Acknowledgements

We wish to thank the referees for the care they put into reading the previous version of this manuscript. Their comments were invaluable in depth and detail, and the current version owes much to their efforts. As usual, however, we alone are responsible for any remaining sins of omission and commission.

Han was supported by the IT R&D program of MKE/IITA 2008-S-024-01 and Salomaa was supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

References

[1] J. Berstel, D. Perrin, Theory of Codes, Academic Press Inc., 1985.
[2] J. Brzozowski, A survey of regular expressions and their applications, IEEE Transactions on Electronic Computers 11 (1962) 324–335.
[3] C. Câmpeanu, K. Culikii II, K. Salomaa, S. Yu, State complexity of basic operations on finite languages, in: Proceedings of WIA'99, in: Lecture Notes in Computer Science, vol. 2214, 2001, pp. 60–70.

[4] C. Câmpeanu, K. Salomaa, S. Yu, Tight lower bound for the state complexity of shuffle of regular languages, Journal of Automata Languages and Combinatorics 7 (3) (2002) 303–310.
[5] M. Domaratzki, State complexity of proportional removals, Journal of Automata Languages and Combinatorics 7 (4) (2002) 455–468.
[6] M. Domaratzki, K. Salomaa, State complexity of shuffle on trajectories, Journal of Automata Languages and Combinatorics 9 (2–3) (2004) 217–232.
[7] Y.-S. Han, K. Salomaa, State complexity of union and intersection of finite languages, International Journal of Foundations of Computer Science 19 (3) (2008) 581–595.
[8] Y.-S. Han, K. Salomaa, D. Wood, State complexity of prefix-free regular languages, in: Proceedings of DCFS'06, 2006, pp. 165–176 (Full version is submitted for publication).
[9] Y.-S. Han, D. Wood, The generalization of generalized automata: Expression automata, International Journal of Foundations of Computer Science 16 (3) (2005) 499–510.
[10] M. Holzer, M. Kutrib, Unary language operations and their nondeterministic state complexity, in: Proceedings of DLT'02, in: Lecture Notes in Computer Science, vol. 2450, 2002, pp. 162–172.
[11] M. Holzer, M. Kutrib, Nondeterministic descriptional complexity of regular languages, International Journal of Foundations of Computer Science 14 (6) (2003) 1087–1102.
[12] J. Hopcroft, J. Ullman, Introduction to Automata Theory Languages and Computation, 2nd ed., Addison-Wesley, Reading, MA, 1979.
[13] M. Hricko, G. Jirásková, A. Szabari, Union and intersection of regular languages and descriptional complexity, in: Proceedings of DCFS'05, 2005, pp. 170–181.
[14] J. Jirásek, G. Jirásková, A. Szabari, State complexity of concatenation and complementation of regular languages, in: Proceedings of CIAA'04, in: Lecture Notes in Computer Science, vol. 3317, 2005, pp. 178–189.
[15] H. Jürgensen, S. Konstantinidis, Codes, in: G. Rozenberg, A. Salomaa (Eds.), Word, Language, Grammar, in: Handbook of Formal Languages, vol. 1, Springer-Verlag, 1997, pp. 511–607.
[16] E.L. Leiss, Succinct representation of regular languages by Boolean automata, Theoretical Computer Science 13 (1981) 323–330.
[17] C. Nicaud, Average state complexity of operations on unary automata, in: Proceedings of MFCS'99, in: Lecture Notes in Computer Science, vol. 1672, 1999, pp. 231–240.
[18] G. Pighizzini, J. Shallit, Unary language operations, state complexity and Jacobsthal's function, International Journal of Foundations of Computer Science 13 (1) (2002) 145–159.
[19] A. Salomaa, D. Wood, S. Yu, On the state complexity of reversals of regular languages, Theoretical Computer Science 320 (2–3) (2004) 315–329.
[20] D. Wood, Theory of Computation, John Wiley & Sons Inc., New York, NY, 1987.
[21] S. Yu, Regular languages, in: G. Rozenberg, A. Salomaa (Eds.), Word, Language, Grammar, in: Handbook of Formal Languages, vol. 1, Springer-Verlag, 1997, pp. 41–110.
[22] S. Yu, State complexity of regular languages, Journal of Automata Languages and Combinatorics 6 (2) (2001) 221–234.
[23] S. Yu, Q. Zhuang, K. Salomaa, The state complexities of some basic operations on regular languages, Theoretical Computer Science 125 (2) (1994) 315–328.